DTIC FILE COPY

(2)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY: UNCLASSIFIED	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited
LE	
R(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 88-0562
AD-A196 942	AFOSR
. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: AFOSR
University of Maryland	
6c. ADDRESS (City, State and ZIP Code): College Park, MD 20742	7b. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR	8b. OFFICE SYMBOL (If applicable): NM
	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR- 82-0303
8c. ADDRESS (City, State and ZIP Code): BLDG #410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO
61102F	2304	A7	

11. TITLE (Include Security Classification): Parallel Logic Programming and ZomB

12. PERSONAL AUTHOR(S): Jack Minker, Mark Weiser

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 6/30/82 TO 2/13/88	88/04/26	11

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify
FIELD	GROUP	SUB. GR.	

DTIC ELECTE JUL 0 7 1988 S D H

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes research performed under a grant to investigate parallel problem solving. A parallel problem sovling system based on logic (PRISM) was designed and implemented on the McMOB parallel processor and ported to a BBN Butterfly parallel processor. This report summarizes the results which were published in nineteen separate reports and papers.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED XX SAME AS RPT [illegible]	UNCLASSIFFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Abraham Waksman	22b. TELEPHONE NUMBER (Include Area Code): 202-767- 5025

DD FORM 1473, 83 APR EDITION [illegible] IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AFOSR-TR-88-0562

FINAL REPORT

AFOSR GRANT 82-0303

NOVEMBER, 1986 - FEBRUARY, 1988

PRINCIPAL INVESTIGATOR:

PROFESSOR JACK MINKER

UNIVERSITY OF MARYLAND

Abstract

This final report presents a summary of research accomplished for the AFOSR under grant number AFOSR 82-0303 for the period November 1986 - February 1988, to investigate parallel problem solving. Under the current grant a parallel problem solving system, PRISM (Parallel Inference System), that was implemented on the VAX/11-780, the PYRAMID and SUN machines, was ported successfully to McMOB and then to the BBN Butterfly parallel architecture. The McMOB architecture is essentially the ZMOB architecture with 16 Motorola 68000 processors, upgrading the Z80A microprocessors, interconnected in a ring structure.

Experimental testing of PRISM on McMOB was undertaken in the current year. In addition, several enhancements were made to PRISM to permit experimental analyses to be made, and to incorporate additional features to take full advantage of parallelism in a problem solving environment. The tracing and statistical gathering packages were extended. An ability to display AND-parallelism was added to the trace program which displays the execution of a program on the parallel machines.

In addition to the above, work continued in the area of informative answers to be presented to a user. Heuristic techniques were developed to determine which information to display.

The system software for ZMOB/McMOB is now robust and considered completed. This has allowed us to reemphasize our studies on parallel software. A new formalism for slicing/splicing was developed which eliminates much of the run-time overhead of the technique, allowing for the development of a splicing compiler. Work has also focused on the development of debugging tools for parallel software and the integration of artificial intelligence techniques into debugging software.

88 7 05 001

1. Introduction

Under the current grant, a detailed design and implementation of a parallel problem solving system based on logic, PRISM (*PaRallel Inference SysteM*), was implemented on the McMOB parallel processor and ported, successfully to the BBN Butterfly parallel processor. The McMOB machine has the identical architecture as ZMOB. It differs in that McMOB has only 16 microprocessors attached to the belt and that these are Motorola 68000 microprocessors. PRISM was previously implemented using a simulated ZMOB belt on VAX, PYRAMID and SUN machines, before it was ported to McMOB. PRISM underwent experimental testing, was enhanced in a number of ways, and a large set of problems was tested using the system.

McMOB was also made operational on the software level. The operating system software for McMOB is now fully functional and no further work is deemed necessary in that area. Further research was performed in the development of program slicers and splicers for automatic parallelization.

In section 2 we provide a description of the accomplishments under the current grant. In the area of parallel problem solving, the initial PRISM has been fully implemented and tested in a parallel environment. Extensive experimentation was begun on evaluating PRISM on a parallel architecture. An approach has been developed to utilize heuristics to obtain informative answers for queries to deductive databases and problem solving systems. In the area of parallel systems hardware and software, the McMOB architecture has been completed, and focus has shifted to the software issues involved. A new formalism has been developed allowing us to exploit regularities in the structures of splicing, drastically reducing the run-time overhead on splicing programs. Further, a prototype for a practical automatic debugging aid, using Artificial Intelligence techniques, has been designed and implemented.

As a consequence of this work we have published 1 journal article, 1 book chapter, 8 conference papers, 2 technical reports, 1 newsletter article, 2 Master of Science scholarly papers, and 2 Ph.D. theses during the present grant period. The list of papers and reports is contained in the section titled, References.

DTIC
COPY
INSPECTED
6

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

2. Accomplishments on Effort During Period November, 1986- February, 1988

This section is subdivided into two major parts. The first section, 2.1, describes the accomplished research with respect to PRISM - the parallel problem solving system. The second section, 2.2, describes the efforts for the development of parallel systems software and hardware for experimentation with parallel algorithms.

2.1. PRISM and Parallel Problem Solving on McMOB and Butterfly

There were seven major tasks in parallel problem solving undertaken under the current grant. These are:

(a) Implement PRISM on McMOB
(b) Application Studies
(c) Alternative Machine Configurations - Design of Combined PSM/IDB Machine
(d) Develop New Features for PRISM
(e) Alternative Architectures
(f) Analyze Parallel Algorithms
(g) Informative Answers
(h) Control Structure Investigation

2.1.1. Implement PRISM on McMOB

We have successfully implemented PRISM on the McMOB parallel machine. The work involved implementing a communications package that allows PRISM programs to communicate over the McMOB belt. All other programs that had been running on the VAX, SUN, and PYRAMID machines were compiled for the McMOB machines and ran successfully. This work is complete, including the implementation of the EDB and Host processors. A paper has been published in the Sigarch Newsletter that describes the system. [Giuliano, Kohli and Minker]

2.1.2. Application Studies

A series of application studies have been performed using PRISM on McMOB. Two problem sets were developed. One was a set of problems known in the literature, and the second was a set of programs generated from abstract trees. The following results have been

obtained: speed ups are generally obtainable when more processors are allocated to solve a problem. Splitting the intensional database component and the problem solving component of the system into separate machines does not appear to be useful. Preliminary indications are that simple heuristics can be used to allocate problems to processors to achieve runtime speed up. Additional experimental work is required before final conclusions can be made on this latter observation. A report has been written that describes results obtained on the PRISM experiments using the McMOB and Butterfly processors (*Experiments with Parallel Logic Programming in PRISM* [Giuliano, et al.])

2.1.3. Alternative Machine Configurations - Design of Combined PSM/IDB Machine

A design for a combined PSM/IDB machine was developed and the design was implemented on McMOB. Due to the studies described above, we believe that there is no need to have separate PSM and IDB machines. Because of this, we have not implemented the EDB communication protocol for the IDB.

2.1.4. Develop New Features for PRISM

The PRISM trace programs were modified to incorporate new features of the system. The trace programs have now been modified so as to display AND-parallel execution and the capabilities of the Constraint Solving Machine (CSM). The trace can run either off-line or on-line with the McMOB system. The user can see the program executing in alternative machines at the same time and can observe both OR-parallelism and AND-parallelism as it is executing on the McMOB. The addition of these features allows us to use the trace to debug programs executing on a parallel machine.

In addition, a list notation and a batch mode were added to the PRISM system. The AND-parallel and OR-parallel PSMs have been combined into one system so that AND/OR-parallelism can be executed simultaneously in PRISM.

2.1.5. Alternative Architectures

We have investigated the BBN Butterfly architecture to determine if it is possible to transfer PRISM to that machine. In contrast to McMOB, the Butterfly machine is a shared memory parallel architecture. We effected a transfer to the Butterfly by simulating the McMob belt. As noted above, this system is operational on the Butterfly and experimental results have been obtained using the system. We have also developed a plan to utilize the shared memory system of the Butterfly and expect to have such a system implemented in the coming grant.

2.1.6. Analyze Parallel Algorithms

A new divide-and-conquer algorithm, called Formula Dissection was developed for propositional satisfiability. The dissection step of the algorithm is implemented by bisecting the underlying graph representation of the formula. The running time of Formula Dissection provides a constructive proof that Planar-3SAT, an NP-Complete class of SAT can be solved in time exp(sqrt(n)) as opposed to exp(n). The Formula Dissection paradigm can also be applied to solve other NP-Complete problems such as finding a Hamiltonian Cycle in a planar graph and the Trihedral Scene Recognition problem in computer vision. A paper on this work was submitted for publication in a journal. [Kasif, Reif and Sherlekar]

2.1.7. Informative Answers

Detailed heuristics have been developed to control the natural language output from deductive databases to provide informative answers to queries posed by a user. A paper has been published that describes some of the heuristics that have been developed. [Gal and Minker]

Implementation of part of this effort has been accomplished. A technical report, [Lobo and Minker], has been written and accepted for publication that describes a meta-program that integrates integrity constraints into a deductive database to semantically constrain the search for answers. The meta-program also has the ability to interface with a relational database.

This basic capability had to be developed prior to obtaining the full capability incorporating the work in the previous paragraph.

2.1.8. *Control Structure Investigation*

A Ph.D. thesis has been written by Madhur Kohli [Kohli] that describes a compiler that permits a user to develop an interpreter with a control capability specified for a particular application. The compiler has been written and experiments have been conducted with the compiler output. An interpreter with the PROLOG control strategy was implemented and compared with PROLOG. Tests run on the VAX machine indicate that the compiled control operates approximately half as fast as PROLOG. However, for control structures that have to be implemented with a meta-interpreter on PROLOG, the compiled interpreter operates approximately ten times as fast as the meta-interpreter. Hence, the approach is both significant and viable for obtaining interpreters with control strategies different than that incorporated in PROLOG. The results obtained are described in Kohli's thesis and in [Kohli and Minker].

2.2. Parallel Hardware and Software

There were four major tasks in parallel hardware and software undertaken under the current grant. These were:

(a) Theoretical Slicing and Splicing
(b) Practical Slicing and Splicing
(c) Automatic Debugging Methods
(d) New Architectures

2.2.1. Theoretical Slicing and Splicing

A major advance in slicing/splicing was made this year in the form of a new methodology for performing the splicing process. The previous method of preventing jumbled outputs, due to processor asynchrony, required resequencing the outputs of different slices and thus each slice was required to continually broadcast certain information to a central location. This

continuous transmission caused a prohibitive expense in terms of overhead for run-time execution, and made the rapid splicing of large programs practically impossible.

The new methodology is based on a formalization of the splicing problem utilizing graph theory. It can be demonstrated from this formalization that certain constraints between outputs are guaranteed to hold and thus, instead of having each splice continually broadcast synchronization information we need send only a small amount of information to a central processor when outputs occur. Tags are attached to the output string which can be deciphered by the coordinating processor and outputs can be guaranteed to be correctly sequenced. These tags are a list of integers bounded by the number of output statements in a slice. We have designed an algorithm which efficiently generates and decodes these tags at run-time. Using the graph theoretic basis of this technique we have been able to provide a proof of the minimality of the average lengths of tags.

The method does not require that the total behavior of a program be represented in its *slices*. Nor does it require that the slices of a program be generated in any particular manner or be implemented in any particular language. It only requires that each slice of a program faithfully reproduce some portion of that program's original behavior. The major advantage of the new method is that it radically reduces the overhead associated with reconstruction of the output [Badger].

2.2.2. Practical slicing and splicing

A major bottle neck in the development of practical slicing/splicing systems has been the run-time overhead associated with providing the synchronization information for the resequencing of output. Our efforts to produce a production quality compiler were stymied by this bottleneck, and work shifted to the theoretical aspects of splicing. As reported above, the new formalization of splicing has broken through that problem, and work is now proceeding on the development of the slicing/splicing compiler for large programs.

2.2.3. Automatic Debugging Methods

Under the current grant, techniques have been explored for applying diagnostic reasoning to program debugging. The difficulty of debugging parallel systems software motivates getting as much automatic aid as possible. Under the current grant we have focused on design and implementation of a prototype system which uses AI techniques to examine control flow information and the values of a variable as the program runs. We have completed a prototype implementation which examines several pieces of LISP code and have shown that the technique will generalize to other interesting classes of bugs. We have also formalized our technique and have demonstrated its validity on a wider variety of bug cases.

Our technique examines various "aspects" of the program as it executes good cases, and compares those aspects with the same program running bad cases. As computation progresses machine learning techniques are used to induce "expectations" of these aspects. An expectation is simply some predicate that satisfies the observed behavior. We then attempt to match the expectation derived from the good cases with the aspect from the anomalous case. If they fail to match, examining this failure provides some insight into the nature of the bug, particularly into identifying where the error has occurred. [Mazurek]

An important aspect of this work has been an effort to make the debugging techniques as language independent as possible. While the learning and diagnostic techniques require AI inferencing capabilities, and are thus best implemented in an AI language such as LISP, the program control-flow analysis does not require that the data be a LISP program. To demonstrate this we have used the multiple language capabilities of the Texas Instruments Explorer machine. These capabilities permit shared memory communications between UNIX and LISP processor boards. We have implemented a version of our prototype which runs on this architecture permitting the LISP processor to monitor the function of programs running on the UNIX board. Using this technique the LISP-based debugger can work on code written in C or other procedural languages. A demonstration of this system will be presented at this year's American Association of Artificial Intelligence Conference in Seattle.

2.2.4. New architectures

The major accomplishment of this grant period was the completion of the McMOB system. The system has now been tested and debugged and the operating system is completely functional. The system is now in use by departmental personnel and support of the machine is being handled by the department.

3. References

1. Badger, L. *Splicing Programs*, Scholarly Paper, Department of Computer Science, University of Maryland, Jan. 1987.

2. Badger L. and Weiser, M. "Minimizing Communication for Synchronizing Parallel Dataflow Programs," *Proceedings of the International Conference on Parallel Processing*, (to appear) July 1988.

3. Callahan, J. and Weiser,M., "Norman Mailer: A Multiple Protocol Mail Reading/Composing Program," IFIP WG 6.5 International Working Conference on MESSAGE HANDLING SYSTEMS (State of the Art and Future Directions), March 1987

4. Gal, A. and Minker, J. "Greater Cooperation between Database and User: Integrity Constraints Provide an Answer," *Proceedings First Annual Conference on Natural Language and Logic Programming*, Vancouver, Canada

5. *Giuliano, M., Kohli, M. & Minker, J. "An Overview of the PRISM Project." *Computer Architecture News*, Volume 15(*1*), March 1987, 35-42.

6. *Giuliano, M., Kohli, M., Minker, J., Rajasekar, A., & Sherlekar, D. *Experiments with Parallel Logic Programming in PRISM* Technical Report Number 1887, Computer Science Department, University of Maryland, July, 1987.

7. Karinthi, R. and Weiser,M., "Techniques of Incremental Execution," *Proceedings, ACM Sigplan Symposium on Interpreters and Interpretive Techniques, SIGPLAN NOTICES 22, 7*, June 1987, 38-44.

8. *Kohli, M. *Controlling the Execution of Logic Programs*, Ph.D. Thesis, Department of Computer Science, University of Maryland, 1987.

9. *Kohli, M. and Minker, J. *Specifying Control for Logic Programs* Technical Report Number 1935, Computer Science Department, University of Maryland, October, 1987.

10. *Lobo, J. and Minker, J. *A Metainterpreter to Semantically Optimize Queries in Deductive Databases*, Technical Report CS-1861, Computer Science Department, University of Maryland, June, 1987 (Accepted for Publication in the Conference on Expert Database Systems, April, 1988).

11. Lyle, J. and Weiser, M., "Automatic Program Bug Location by Program Slicing," Conference on Computers and Applications, Peking, Peoples Republic of China, July 1987.

12. Mazurek, M. *Expectation Formation and Its Use in Debugging*, Proposed PhD Thesis, Computer Science Department, University of Maryland, 1987.

13. Mazurek, M. *Towards the Automatic Parallelization of Sequential Programs*,, Scholarly Paper, Computer Science Department, University of Maryland, 1987.

14. Minker, J. "Perspectives in Deductive Databases," *Proceedings of the 6th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, (Abstract of Invited Talk) San Diego, Cal., March 1987.

15. *Minker, J. "Perspectives in Deductive Databases," *Journal of Logic Programming, 1988:5*, 33-60.

16. *Sherlekar, D. *Graph Separator-Based Techniques in VLSI and Algorithms*, Ph.D. Thesis, Department of Computer Science, University of Maryland, 1987.

17. Weiser, M and Badger, L., "Automatic Detection and Use of Process Parallelism," (in preparation).

18. Weiser, M. "Source Code!," *IEEE Computer*, November 1987, 66-74.

19. Weiser, M. and Shneiderman, B., "Human Factors of Software Design and Development," in *Handbook of Human Factors*, ed. Gavriel Salvendy, John Wiley & Sons, 1987.